Theses and Dissertations	1. Thesis and Dissertation Collection, all items

1983

The application of a general purpose data base management system to design automation.

Walden, Heather J.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/19643

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE APPLICATION OF A GENERAL PURPOSE
DATA BASE MANAGEMENT SYSTEM
TO DESIGN AUTOMATION

by

Heather J. Walden

December 1983

Thesis Advisor: A. A. Ross

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED
The Application of a General Purpose Data Base Management System to Design Automation	Master's Thesis; December 1983
	6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)
Heather J. Walden	

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Naval Postgraduate School Monterey, California 93943	

11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE
Naval Postgraduate School Monterey, California 93943	December 1983
	13. NUMBER OF PAGES
	150

14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office)	15. SECURITY CLASS. (of this report)
	UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer-Aided Design, Oracle, General Purpose Relational DBMS, Relational Model for a CAD System, Computer System Design Environment (CSDE)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis describes the analysis of the Computer Systems Design Environment data base requirements, the design of a relational model to fulfill those requirements and the implementation of that model on a general purpose data base management system.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

1

The Application of a General Purpose
Data Base Management System
to Design Automation

by

Heather J. Walden
Lieutenant, United States Navy
B.S., Penn State University, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1983

ABSTRACT

This thesis describes the analysis of the Computer Systems Design Environment data base requirements, the design of a relational model to fulfill those requirements and the implementation of that model on a general purpose data base management system.

TABLE OF CONTENTS

I. INTRODUCTION

The goal of this research was to determine the validity
of the hypothesis that a general purpose relational data
base management tool could be used to implement a computer
aided design (CAD) system, and, if found valid, to provide
the design of such a data base.

Computers have been used to aid in the design of a
great number of things, the most widely recognized being VLSI
chips. The design of VLSI chips is related to the purpose
of the work supported here. Its purpose is to incorporate
some previously defined function into VLSI technology. This
system, the Computer System Design Environment (CSDE),
instead of defining a new chip uses existing microprocessor
technology to devise a specific controller for a given
design problem. For instance, suppose one needs a microwave
oven controller. Given the performance parameters, this
system can design the software, hardware and interconnections
of a microprocessor system to perform this control. In
other words, this computer-aided design system is used to
map a functional specification to a microprocessor and its
associated software and peripherals.

This thesis work is an extension of the work done by
Alan Ross in his Ph.D. dissertation, "Computer Aided Design

Microprocessor-Based Controlers",. His dissertation

scribes the demonstration of the feasability of automati-

lly designing microprocessor-based real-time systems.

iefly, his method was to: 1) create a problem statement

unctional specification), 2) translate it into an inter-

diate format, 3) select a microprocessor realization from

library of realizations, 4) generate the software and the

rdware to implement the function on the chosen micropro-

ssor, 5) check the timing constraints and develop a

nitor and 6) output the design description. [Ref. 1]

One of the shortcomings of his work is an overly complex

plementation because he did not use an adequate data base

ol to support the library of microprocessor realizations.

s system maintained a 'data base' on formatted, IBM

nched cards. That method of storage is highly inflexible

d resistant to change. It was at best a bulky, awkward

y of storing and carrying around information. Times have

anged and with the aid of currently available data base

nagement tools, new implementations can be considered.

One such new tool is Oracle: a general relational data

se management system. It is available on the Vax 11/780

ler the VMS operating system and its command language is

. (pronounced sequel). A specific goal of this thesis was

identify Oracle's usefulness in designing a relational

el of the data base requirements of the CSDE.

II. PROBLEM DEFINITION

In designing a data base for any system one must make a careful and dedicated analysis of the system's requirements, its overall function and its design features. Secondarily, it is also necessary to discern the system's limitations with respect to the availability of technological, funding and manpower resources at the time of conception. It is also a good idea to study the system's goals for the future and plan for extensibility. By doing this, one discovers the data that is necessary for each function and the relationships the data have for each other. With this information a data base schema can be drawn up that maximizes the system's capabilities and performance. The system under scrutiny here is a computer-aided design (CAD) system.

In general, a CAD system is required to automate a certain function, whether it be the design and manufacture of microprocessor ships or something totally unrelated to the computer industry. One instance of a CAD system is the Structured Computer Aided Logic Design (SCALD) system, developed out of Lawrence Livermore Laboratories. It operates by mapping the designer's input text and graphics-based instructions to a technology specific wire-wrap list for processor board construction. In performing this

mapping, the SCALD system frees the designer from mechanical, repetitive tasks that are tedious and error prone for human beings. It allows the designer to spend his/her time creatively rather than robotically. [Ref. 2]

The function and operation of the CSDE is very similar to the SCALD system. A design problem is entered into the CSDE in the form of a high level language called the Computer System Design Language (CSDL). The CSDE then attempts to map the design criteria outlined in the CSDL to a specific microprocessor system.

For the target system of this research, the Computer System Design Environment (CSDE), the communication lines between the data base and the rest of the CSDE system are related in Figure 1. The user has access to the data base for three operations: 1) to input a problem statement, 2) to input microprocessor realizations and 3) to manipulate design configurations.

For problem statement input, the user/designer interface is based on a syntax directed editor (SDE) [Ref. 3]. The problem statement which is produced by the SDE includes the following sections:

   (1)   identification section - contains user identification and documents the problem for record keeping purposes.

   (2)   environment section - defines the interface between the controller and the process to be controlled; lists the external and internal variables which are defined for the controller.

USER

DATA BASE

PROBLEM
STATEMENT
INPUT

MICROPROCESSOR
REALIZATION
LIBRARY

FINAL
DESIGN
CONFIGURATION

TRANSLATED
TABLES

ENVIRONMENT
DESIGN CRIT.
ID
PRMTIV LST
TIMING

FORMATTER

FUNCTIONAL
MAPPER

OPTIMIZER

REALIZATION
TIMING
TABLE
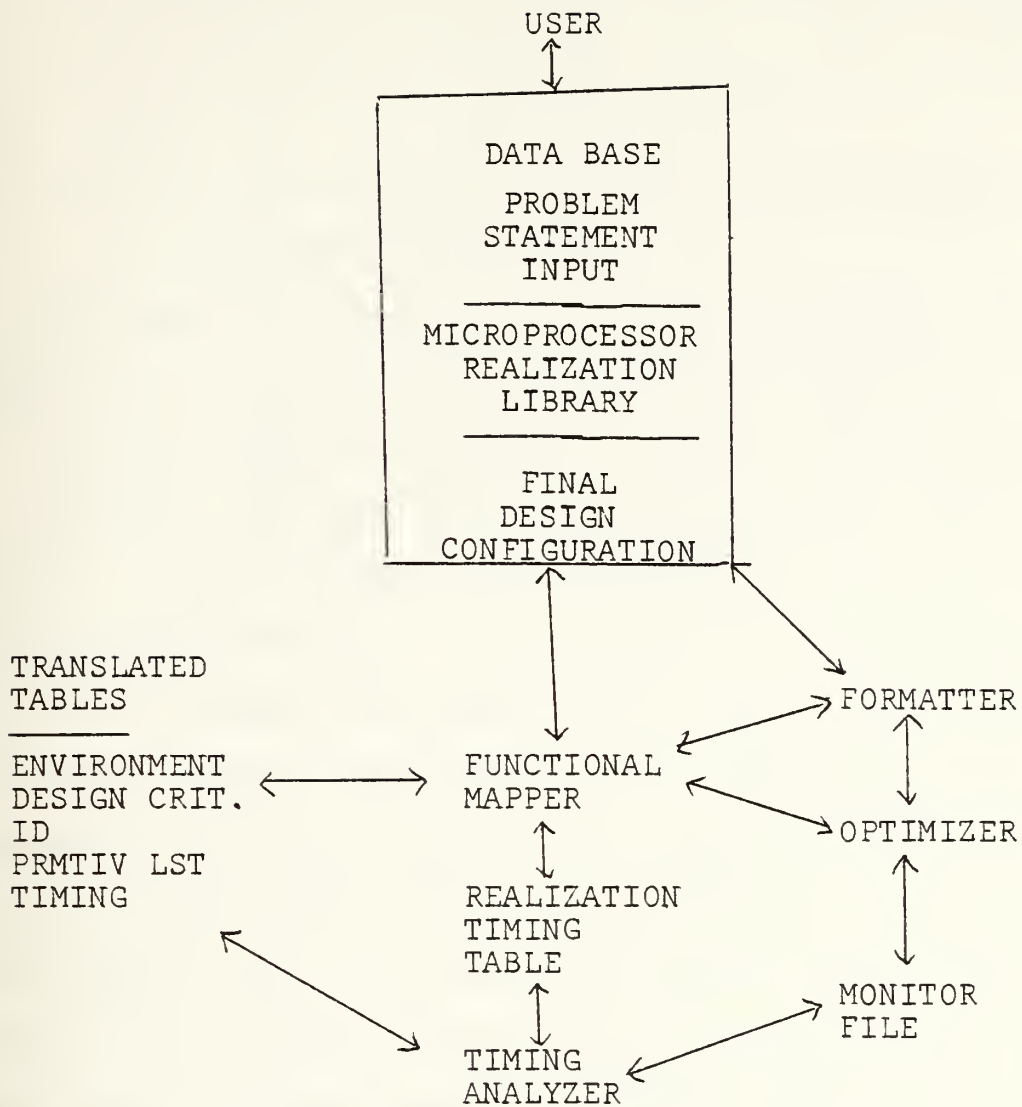
MONITOR
FILE

TIMING
ANALYZER

Figure 1 Original CSDE Layout

10

(3)   contingency list – list of the contingencies (stimuli
       the controller must respond to) and tasks (controller
       responses) and their time constraints.

(4)   procedures section – details the routines which define
       the contingency/task pairs.

An example of a problem statement input for a fuel injection controller is shown in Figure 2 [Ref. 4].

In the design of the current system as shown in Figure 1, a Translator uses the information in the format above to create a primitive list and a symbol table which are used by the Functional Mapper. The Translator is not yet designed but is an ongoing development in the CSDE project. The primitive list is developed from the contingency list and procedures section of the problem input statement. The symbol table contains the attributes for the variables in the environment table. The Functional Mapper tries to find the software and hardware macros listed in the primitive list by searching a microprocessor realization selected from the data base library. If there is a functional match, the Functional Mapper uses the symbol table to match the attibutes of the primitive's variables to the attributes of the arguments in the realization.

During this process, the Functional Mapper produces a realization timing table which contains the accumulated time durations of the enlisted macros. The Timing Analyzer then

IDENTIFICATION:

  Designer: Pollock/Loomis
  Date: 23-Jun-82
  Project: 280Z Fuel Injection Controller

ENVIRONMENT:

  Input: IF,8,RESTRAN;THR,1,TTL;THF,1,TTL;AT3,8,TEMP;
          ATR,1,TTL;WTR,1,TTL;WT3,8,TEMP;S,1,SWITCH;
          STSW,1,TTL;
  Output: W,8,TTL;
  Arithmetic: KA,24:REAL; K1,24:REAL; ALT,24:REAL; WT,24:REAL;
               AT,24:REAL; TH,24:REAL; OLDS,1:BINARY;

CONTINGENCY LIST:

  SECTION: INITIAL;

    "Initialization Contingency"
    when START:100MS do SETUP;

  end "INITIAL";

  SECTION: RUN;

    "Control Fuel pulse width contingency"
    when CW:35MS do TW;

    "Altitude, Water Temp and Air temp factors contingency"
    when CKA:100MS do TKA;

    "Throttle sensing contingency"
    when CTH:200MS do TTH;

    "Air temp factor contingency"
    "AT=(1+(20-T)/400)"
    when CATMP:240 do TATMP;

    "Water temp factor contingency"
    "WT=WT1+(70-T)**2/1E4"
    when CWTMP:240 do TWTMP;

    "Altitude correction factor contingency"
    "If p<26 inHg then ALT:=1 else ALT:=.94"
    when CALT:240 do TALT;

end "RUN";

    Figure 2 CSDL Description of Fuel Injection Controller

```
PROCEDURES;
  function START;
    binary START,1;
    sense STSW;
    if STSW=1 then START:=1 else START:=0;
  exit START;

  function CW;
    binary CW,1;
    CW:=1;
  exit CW;
    .
    .
    .
  Function CALT;
    binary CALT,1;
    sense S;
    if S/=OLDS then CALT:=1 else CALT:=0;
  exit CALT;

  task SETUP;
    kl=10.5;
    do TTH;
    do TATMP;
    do TWTMP;
    do TALT;
    do TKA;
    switch section to RUN;
  exit SETUP;

  task TW;
    sense (IF);  "resistive transducer for air flow"
    W:=IF*KA*Kl;
    issue W;
  exit TW;

  task TKA;
    KA:=ALT*WT*AT*TH;
  exit TKA;
    .
    .
    .
  task TALT;
    sense (S)      "altitude switch"
    OLDS:=S;
    if S=1
      then ALT:=.94
      else ALT:=1.
    fi;
  exit TALT;
```

<div align="center">Figure 2   continued</div>

compares the realization timing table to the application timing table to ensure that user specified timing constraints have been met. The Timing Analyzer also adds to the set of software and hardware macros in order to implement a monitor scheme for the application.

Finally, results are ready for output by the process Formatter. If the user has specified that another realization in the library be tested for feasibility, the Functional Mapper is again accessed and the process is repeated. The Optimizer can then select, again based on user specification, the optimal solution and have it formatted for output or display.

The above is an overview of the data flow as it existed prior to this research effort. Modifications have been made in this process to incorporate Oracle and to enhance its efficiency. These and other details are addressed in the design and implementation chapters.

The second required operation, the entering of microprocessor realizations, is done through the Application Design Interface (ADI) developed as a part of this thesis effort with the use of Oracle's Application Interface Facility (AIF). The AIF is helpful because it permits the screen interface designer to set up strong data typing to help enforce correct data entry. After the designer has created the data base tables and types, the ADI (a screen oriented, user-friendly system) prompts the user for

microprocessor(s), the ADI allows the designer to instan-
tiate an Oracle version of it by entering data through a CRT
terminal interface. In this way, microprocessor realizations
are stored in the data base for later use by the CAD system.
[Ref. 5]

The third area of user interaction involves the Final
Design Configuration (FDC). Once the design of a particular
controller is final, a designer may wish to look at certain
statistics concerning its makeup. Statistics are generated
according to specifications in the Design Criteria Table
established by the designer upon problem statement input.
These statistics are automatically stored in the data base
after each successful run of the CAD system. The statistics
include chip count, power requirement, monitor scheme,
storage, time, macro index and chip list.

For the design of a microprocessor-based controller,
the CAD system must use the data base to map the problem
statement to a set of software and hardware macros and to
display the solution to the designer. Returning to Figure 1,
the areas of the existing program that interfact directly
with the data base will have to be rewritten to incorporate
the use of a relational data base management tool.

The CSDE system is also not fully developed or
designed. It is limited to the generation of real-time
controllers and develops non-preemptive scheduling schemes.
Although its basic function has been proven feasible,

15

expanded capabilities for improved design flexibility, such as graphics display, have not been created. All of these undeveloped and undesigned CAD features impact the design of a data base that should be easily extendable to support them.

Although efforts are underway to develop the Z80 and 8086 specifications, currently, the 8080 microprocessor is the only realization ready for implementation into the data base. Other microprocessors will be added to the library as they become available. Therefore, the data base must provide a representation that is general and readily extensible. It must also provide a means to contain all the idiosyncracies of an individual microprocessor without impacting other parts of the system.

There is currently only one monitor scheme implemented. This is another factor whose attributes must be examined in designing the data base, but this fact must not change the data base representation just to suit its needs specifically.

In reviewing the above analysis of the CSDE, 3 basic guidelines concerning any remodeling of the current data base design can be detected. First of all, the design should enhance the designer's ability to do his/her job. The three user interfaces - problem design, microprocessor volume design and final controller design - should all be easy to understand, simple to use and increase the designer's potential performance. Second, the redesign of

the data base should not complicate the interworkings of the CSDE. If possible the redesign should help streamline the operation of the CSDE and unravel any 'clever' programming schemes that were built in out of necessity due to working with the Fortran/IBM punched card format. And third, the data base design should be flexible enough to incorporate future changes readily. The design should apply the well-known software engineering principles of modularity, information hiding and regularity. The system should be designed so that the information is easily and logically encapsulated within the framework of the data base, but yet its implementation should be invisible to the user. The use of all three principles enhances changeability and, as already presented, this CAD system is and will be in a state of change.

The next chapter describes a new design for the CSDE using a relational data base management system. It addresses the organization of data within the data base and possible implementation schemas for use by the CSDE subroutines.

III. <u>DESIGN</u>

The last two chapters explained the requirements and rationale of the CSDE. The purpose of this chapter is to describe the design of a data base for the CSDE that fulfills those requirements. This design chapter is primarily concerned with the arrangement of the CSDE's nece1sary data into logical entities that can be manipulated by a relational DBMS, specifically Oracle. This specific design is not directly transferable to another type of DBMS. However, an effort has been made to encapsulate the important operations and ideas so that the resulting design can easily be modified or rehosted on a different DBMS.

As noted in Chapter II, there are three main conceptual areas which must be served by the data base for the CSDE. These can be classified as: 1) problem statement input, 2) microprocessor realization library and 3) the final design configuration for a given controller. The data base has been organized around these three data entities. Figure 3 illustrates the relationships that the major subroutines of the CSDE have with these three areas of functionality.

The user enters the problem statement via the syntax directed editor (SDE). The information required by the Functional Mapper (the primitive list and the symbol table)
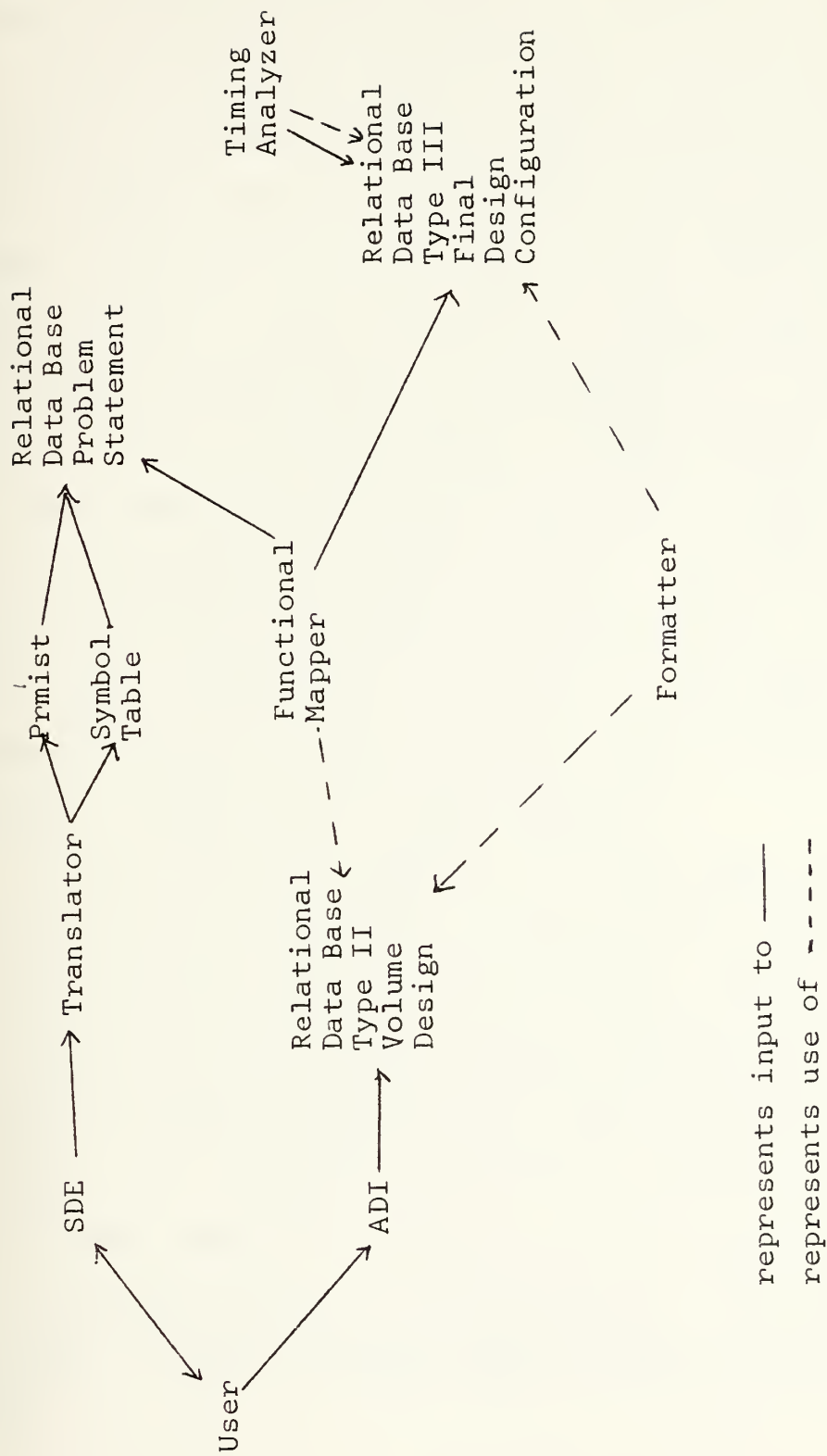
Figure 3 CSDE Subroutine Relationships

19

is then extracted by the Translator and is stored in what is termed a relational data base type 1 (RDBT1). The remaining information from the problem statement input - the problem statement identification, the design criteria and the environmental criteria (in the form of the application timing table) - are stored in the final design configuration or RDBT3. RDBT3 is meant to be both a working environment and a historical repository for the designer. It therefore con-trains all the necessary facts to extract useful design abstractions such as chip count and power supply requirements and to reconstruct the design effort. The RDBT3 will also contain the set of information comprising the current realization timing table and a list of the contingencies and tasks of the design problem with an index to their specifically required software and hardware macros.

As can be seen by Figure 3, the data base must also hold microprocessor realizations - the primitive implementations which are used to construct the designs. These are contained in RDBT2s and are entered through an Application Design Interface created using Oracle's Application Interface Facility. The relational model must therefore also support the volume designer's effort and supply the fields necessary to adequately describe a microprocessor. The following pages describe in detail the relational models for each data base type.

The first relational data base type - RDBT1 - is created
from information extracted from the initial CSDL problem
statement input by the translator. This information is in
the form of a primitive list and a symbol table. The
primitive list is generated from the contingency list and
procedures sections and the symbol table is derived from the
environment section of the problem input. The primitive
list contains the titles of each function required by the
given problem, the primitives (or software macros) that
produce each function and the arguments or parameter vari-
ables necessary for each primitive. The symbol table
contains the set of actual parameters used in the given
problem statement with their attributes and precisions.

The RDBT1 is created by a module - CREATEDB1 - that is
invoked by the Translator module. If it is decided to
change the data base management system, the CREATEDB1 module
will be the only section to require major modification to
incorporate the change. The RDBT1 will also be accessed by
the Functional Mapper to create a mapping of RDBT1 to an
RDBT2 contained in the realization library. The resulting
mapping will be stored in an RDBT3 as aprt of the final
design configuration.

The relational model is simple and mirrors and contents
of the primitive list and symbol table in a modified
relational form. The model is shown in Figure 4.

PRIMITIVE_LIST

field	TITLE	MACRONAME	POINTER_X
type	CHAR	CHAR	NUMBER

SYMBOL_TABLE

field	POINTER_X	ARGUMENT	MINIMUM	MAXIMUM
type	NUMBER	CHAR	NUMBER	NUMBER

Figure 4   Translator Output

Title is a primary key to the first table in this section of the data base. Title is the name of a contingency or task defined in the original CSDL input by the designer. Macroname is a software macro which is also selected as a part of a contingency or task in CSDL input. Pointer_x/Argument make up the key for the second table. Pointer_x is used as a 'pointer' from the table Symbol Table(s) to the Primitive List or in other words, Pointer_x is the relation that binds the two tables together. Pointer_x is a unique number generated for each Title/Macroname relation supplied by the Translator.

The second table is used to hold the actual parameters and their precision requirements as defined in the CSDL description of a problem. These are represented in the second table by the fields Argument, Minimum and Maximum. In data base terminology, the relationship between Table 2 and Table 1 is many to one. In this way, the relational model avoids repetition of the contents of the symbol table and provides a data base that is easy to comprehend and logically oriented. Figure 5 shows the recommended arrangement of data fields for 2 simple tasks.

The second functional area of the CSDE which the data base supports is the microprocessor realization library. Each microprocessor realization in the library is represented as a volume. Each volume is further subdivided into two more sections; hardware and software. It is from

23

PRIMITIVE_LIST

TITLE	MACRONAME	POINTER_X
TASK1	S.SENSECOND	1
TASK1	S.ADD	2
TASK2	S.MULT	3

SYMBOL_TABLE

POINTER_X	ARGUMENT	MINIMUM	MAXIMUM
1	SIGNAM	0	8
2	ARG1	0	8
2	ARG2	0	8
2	RSLT	0	8
3	PARAM1	0	8
3	PARAM2	0	8
3	ANSWER	0	8

Figure 5   Relational Model for a Simple Contingency/
Task Function

24

these volumes of hardware and software components that the CSDE attempts to build a microprocessor controller complete with monitor for a given problem specification.

Each hardware and software primitive in a volume has specific characteristics that must be identified by the volume designer. These characteristics fall into three general categories that easily lend themselves to a relational model. The categories are functional specifications, control lines and text blocks.

Since the hardware and software primitive have different characteristics, two different type of tables have been devised. The two table types are also differentiated into blocks that represent one of three relational categories. These three categories are the functional specification, the control lines and the text code. These categories are further defined in the following pages.

In any relational model, a separate block or table is needed to define multiple instances of a field. In other words, a particular block can have no two fields the same. For instance, a hardware primitive may have more than 1 input parameter, so a separate block is necessary to input parameters. Several of the input parameters in this block will be incorporated into the design of each hardware primitive. The tables, blocks, fields and field types are displayed in Figure 6 and 7 for the hardware and software macros, respectively.

```
                    BLOCK 1 - IDENTITY_BLOCK

field   MICROPROC   HW_MACRO   MILLIWATTS   CHIPCOUNT   LATENCY
----------------------------------------------------------------
type    CHAR        CHAR        NUMBER       NUMBER      NUMBER


field   POINTER
---------------
type    NUBMER

                  BLOCK 2 - PARAMETER_BLOCK

field   POINTER   PARAMETER      MIN      MAX
-------------------------------------------------
type    NUMBER    CHAR           NUMBER   NUMBER


                  BLOCK 3 - COMMENT_BLOCK

field   POINTER   COMMENT   COM_ID
-----------------------------------
type    NUMBER    CHAR      NUMBER


                  BLOCK 4 - INCLUDE_BLOCK

field   POINTER   INCLUDE   INC_ID   TIMES
-------------------------------------------
type    NUMBER    CHAR      NUMBER   NUMBER


                  BLOCK 5 - CALL_BLOCK

field   POINTER   CALL   CALL_ID   TIMES
-----------------------------------------
type    NUMBER    CHAR   NUMBER    NUMBER


                  BLOCK 6 - CALC_BLOCK

field   POINTER   GLOBVAR   OP    VAR    RSLT   CALC_ID
--------------------------------------------------------
type    NUMBER    CHAR      CHAR  CHAR   CHAR   NUMBER
```

Figure 6   Hardware Macro Table

BLOCK 7 - ATTRIBUTE_BLOCK

field	POINTER	LOCALVAR	OP	VAR	RSLT	ATTR_ID
type	NUMBER	CHAR	CHAR	CHAR	CHAR	NUMBER

BLOCK 8 - IF_BLOCK

field	POINTER	VARIABL1	OP	VARIABLE2
type	NUMBER	CHAR	CHAR	CHAR

field	IF_MP	IF_HWMAC	IF_REQ	IF_ID
type	CHAR	CHAR	CHAR	NUMBER

BLOCK 9 - TEXT_BLOCK

field	POINTER	TEST	TEXT_ID
type	NUMBER	CHAR	NUMBER

Figure 6   Continued

27

```
                BLOCK 1 - IDENTITY_BLOCK

field   MICROPROC   S/W_MACOR   STORAGE   TIME    EXT_REF
----------------------------------------------------------
type    CHAR        CHAR        NUMBER    NUMBER NUMBER

field   POINTER
---------------
type    NUMBER


                BLOCK 2 - ARGUMENT_BLOCK

field   POINTER   ARGUMENT   MINIMUM   MAXIMUM
-----------------------------------------------------
type    NUMBER    CHAR       NUMBER    NUMBER


                BLOCK 3 - COMMENT_BLOCK

field   POINTER   COMMENT   ID
-------------------------------------
type    NUMBER    CHAR      NUMBER


                BLOCK 4 - INCLUDE_BLOCK

field   POINTER   INCLUDE   ID      TIMES
-----------------------------------------------
type    NUMBER    CHAR      NUMBER  NUMBER


                BLOCK 5 - CALL_BLOCK

field   POINTER   CALL   ID      TIMES
-----------------------------------------
type    NUMBER    CHAR   NUMBER  NUMBER
```

Figure 7   Software Macro Table

BLOCK 6 - CALC_BLOCK

field	POINTER	GLOBVAR	OP	VAR	RESLT	ID
type	NUMBER	CHAR	CHAR	CHAR	CHAR	NUMBER

BLOCK 7 - ATTRIBUTE_BLOCK

field	POINTER	LOCALVAR	OP	VAR	RESLT	ID
type	NUMBER	CHAR	CHAR	CHAR	CHAR	NUMBER

BLOCK 8 - IF_BLOCK

field	POINTER	VARIABLE1	OP	VARIABLE2
type	NUMBER	CHAR	CHAR	CHAR

field	IF_MP	IF_MACRO	IF_REQ	IF_ID	ID
type	CHAR	CHAR	CHAR	NUMBER	NUMBER

BLOCK 9 - TEXT_BLOCK

field	POINTER	TEST	ID
type	NUMBER	CHAR	NUMBER

Figure 7   Continued

.

29

In the HARDWARE_MACRO table there are 9 block types. The first two blocks represent the functional specification of the hardware primitive. The information in the first block is required for every hardware primitive. It has two identifying keys: 1) the microprocessor and 2) the name of the hardware primitive. A unique number is then generated when the volume is designed, to as as a pointer to identify any of the remaining block types associated with the particular primitive. The other fields in the first block contain information required by the CSDE to compile power, timing and chip count requirements for the controller.

The next block is used to hold the hardware primitive's parameters or arguments. A hardware primitive may have 0, 1 or as many parameters as it requires. Each argument must also be identified by its minimum and maximum precision. For instance, in the example used in reference , the hardware primitive H.SENSECOND for the 8080 requires an argument called SIGNAM whose minimum precision is 0 bits and whose maximum precision is 8 bits. The Functional Mapper will use the information in this second block to try and match each primitive from the problem statement input with a realization from the volume.

The next 7 blocks in the HARDWARE_MACRO table represent the control information needed to design a primitive. The first control type is the COMMENT_BLOCK. The COMMENT_BLOCK is included in the realization volume as an aid to the

volume writer in keeping track of the purpose for and operation of the realization primitive. Comments can be 80 alphanumeric characters long in keeping with the general width of a page. The zero, one, infinity rule applies to the number of COMMENT-BLOCKs permitted as well as to the rest of the control line blocks. Each comment line is given an identification number generated by the DBMS. This is to keep things organized for output of a realization volume. Comments are not acknowledged by the Formatter and are, therefore, not output in the Final Design Configuration (FDC).

There are two different types of outputs which must be generated from each realization volume. The first output is simply a designer tool; a user manual or guide reflecting the hardware and software primitives of a particular microprocessor volume. The second output is the FDC of a microprocessor controller for a specific design problem.

The next two blocks represent similar control functions. Sometimes it is necessary for a primitive to use another primitive within its scope. When an INCLUDE_BLOCK is included in a primitive's format the Formatter will include the primitive in the FDC but list it separately, after all the initially indexed primitives have been output. The CALL_BLOCK tells the Formatter to include a primitive in the listing at the point where it was called. Each of these blocks are identified by a system generated identification number. These identification numbers not only help to

organize the realization volume output but become essential when implementing conditional (IF) control lines that incorporate CALL and INCLUDE functions. The last field, labeled TIMES, is unique to these two control blocks and can be used by the volume writer to implement a controlled loop structure similar to the case type structure implemented in reference . Its use is explained later in the section describing the IF_BLOCK.

The CALC_BLOCK and the ATTRIBUTE_BLOCK also have similar form and function. Both are used to assign values to system variables. The CALC_BLOCK is included in the design primitive to allow the manipulation of design system global variables. During system generation, values are assigned to the global variables by subroutines called by the Formatter. An example of this in the 8080 microprocessor realization library is the Portpt global, which stores the number of I/O ports required for a design, so the system can assign a specific address to each I/O port. The expression which calculates the global variable is represented as four fields in the relational model; one field to represent the actual global used, one field for the arithmetic operator (*, **, +, -, /), one field for the value applied against the operator toward the global and one field to hold the result. Complicated expressions that would normally involve parentheses will have to be simplified and distributed over

more than one CALC_BLOCK. The ID field provides a mechanism for ordering the expressions.

An example strategy, using the relational model in the design of a microprocessor primitive is given in Figure 8. The expression Total =)Global_add + Const) * Global_mult is divided up into two expressions. The result of the first expression, SUM, is used to provide a common link between the expressions. The ATTRIBUTE_BLOCK contains the same field types as the CALC_BLOCK. It is used to assign values to local variables within the primitive in the same manner in which the CALC_BLOCK operates.

The IF_BLOCK is a feature which allows the volume designer some additional flexibility. The IF_BLOCK represents a cross between a case statement and a DO FOR I = 1 to X loop. The IF_BLOCK originally had the following syntax:

IF <math-exp> . <rel-op> . <math-exp> SKIP <lines>

The SKIP <lines> clause was employed as a method to include or not to include lines of the primitive. For example, upon system generation the global variable representing the total amount of additional RAM required by a functional specifica- tion might equal 1792. In this case, 7 additiona 256 byte RAM primitives are needed to fulfill the functional specifi- cation. Instead of calling the H.RAM256 macro 7 times, an if statement had been incorporated into the H.RAM256

POINTER	GLOBVAR	OP	VAR	RESLT	CALC_ID
1	GLOBAL_ADD	+	CONST	SUM	1

POINTER	GLOBVAR	OP	VAR	RESLT	CALC_ID
1	GLOBAL_MULT	*	SUM	TOTAL	2

Figure 8   Example Microprocessor Primitive Design

hardware primitive allowing for the inclusion of additional RAM up to the maximum allowed by the 8080 hardware set.

The IF_BLOCK of the relational model allows for the same type of primitive implementation technique. The syntax of the IF statement has been changed to the following:

IF <var> . <rel-op> . <var> THEN <call or include>

The call or include statement will then contain the number of times the primitive is to be called or included according to the relationship identified between teh variables in the IF statement. A possible implementation of the previous example would be:

IF <memrept> .GT. <1791> THEN call (8080.H.RAM256.7)

As in a case statement each possible value for <memrept> can be represented in the primitive. 8080 corresponds to the IF_MP field, HRAM256 corresponds to the IF_MACRO field and 7 corresponds to the IF_REQ field. During a CSDE system run, the IF_REQ field requirement will be identified and this number can then be placed in the TIMES field of the appropriate CALL_BLOCK or ATTRIBUTE_BLOCK. During system generation, the Formatter will know haw many times to include a particular primitive by the quantity held in this field. An IF_ID field is then generated by the system to keep the IF statements in order for the final outout listing. The IF_BLOCK offers the volume designer flexibility

and an avenue for creativity without complicating the design of the relational representation too much.

The final functional category in the microprocessor volume design is the TEXT_BLOCK. The TEXT_BLOCK contains assembly level code for the microprocessor. Each line of the code is identified by the ID field to keep the code in order. Without using some type of incremental function in an extra field on which the DBMS can sort there would be no way of identifying which line came first in the coding specification.

Everything contained in the TEXT_BLOCKS will be output in the final design as is expect for two types of text. The first type is a variable enclosed in pound signs (#). These will be interpreted as calls to a system procedure of function, and the corresponding procedure or function will be implemented to generate character strings for the final design listing. The two system functions which are currently implemented are: 1) 'idsec' which prints the next line of the ID table and 2) 'status(n)' which returns status information on the storage attributes. The second exception is the handling of dummy arguments. Within the text lines, each dummy argument is enclosed in brackets ('<' and '>') and are replaced by the actual arguments wherever the '<' and '>' are encountered.

The SOFTWARE_MACRO table has the same three functional categories represented by its relational model. The only difference is in the first block containing the primitive's

36

functional specifications. Whereas the Hardware primitive
required fields to allocate latency, timing and power
requirements the software primitive requires fields for
storage, timing and external memory references. These
fields will be used by the CSDE to calculate total storage
and time requirements for the microprocessor system and to
develop a monitor.

These first two data base types - the Relational Data
Base Type 1 (RDBT1) containing the primitive list and the
symbol table and teh Relational Data Base Type 2 (RDBT2)
containing the realization volume design - will be used by
the Functional Mapper to build a major portion of the final
design configuration (RDBT3). The current CSDE accomplishes
the mapping in two separate operations. First, the index
of the chosen realization volume is checked for a match to
a given design primitive. If a match is found, the
selection criteria in the design primitive are compared
to selection criteria in the realization volume primitive.
By using a relational model, this process can be shortened
into one operation. The SQL language will allow a SELECT
operation on the data base to retrieve a match to any and
all parameters requested. So in a programming procedure a
controlled loop can be created whereby design primitives
with their arguments and precision requirements are fed into
dummy parameters of a SQL select statement. This select
statement will then be the means by which the RDBT2 is
searched for a match.

Figure 9 shows a demonstration of this idea. The procedure, FUNMAP, is called from the Optimizer which sends as input variables the MACRONAME - name of the design primitive to be matched to the macro in the realization library - ARGUMENT - one of the design primitives I/O parameters - and MIN, MAX - the precision requirements of the ARGUMENT sent. As is evident, in this scheme the Funmap procedure is invoked to test each argument against a possible match in the realization libaray. The Funmap procedure then sets a boolean variable - FOUND - equal to tru or false depending of whether a match was found or not. Other schemes could be drawn up placing more responsibility on the Funmap design. For instance, it could be argued that it is the responsibility of the Funmap to place the found primitive in the PRIMARY_ INDEX in the Final Design Configuration. The purpose of this figurative Funmap was not to dictate its future but to show how the SELECT statement streamlines the mapping process.

The third and final functional area of the CSDE requireing data base support is the Final Design Configuration (FDC). As noted previously the RDC is to be used as a working environment and as a historical notebook for the designer. It should provide enough information about the microprocessor design to discern useful patterns and abstractions about the design and to be able to recreate the designer's original goals in using the CSDE.

```
Procedure Funmap (NACRONAME, ARGUMENT, MIN, MAX, FOUND);

   SELECT from IDENTITY_BLOCK

      where (MICROPROC.SW_MACRO = MACRONAME
      and    MICROPROC.SW_MACRO.ARGUMENT = ARGUMENT
      and    MICROPROC.SW_MACRO.ARGUMENT.MINIMUM = MIN
      and    MICROPROC.SW_MACRO.ARGUMENT.MAXIMUM = MAX)


      or    (MICROPROC.HW_MACRO = MACRONAME
      and    MICROPROC.HW_MACRO.ARGUMENT = ARGUMENT
      and    MICROPROC.HW_MACRO.ARGUMENT.MINIMUM = MIN
      and    MICROPROC.HW_MACRO.ARGUMENT.MAXIMUM = MAX) ;


If SELECT then
      FOUND = TRUE
Else
      FOUND = FALSE;

end Funmap;
```

Figure 9   Functional Mapper Demonstration

There are four primary functional areas in the FDC.
They are the:

    (1)   Identification Section

    (2)   Design Criteria Section

    (3)   Timing Tables

    (4)   An index of the hardware and software primitives
        used in the design.

The first area is an identifying key to the FDC. The
information in the ID section is input by the designer with
the Syntax Directed Editor (SDE) at the beginning of the
design process. The data required by the relation includes:
1) the designer's name, 2) the date, 3) the name of the
design project and 4) version number. Each identification
table will also contain an extra field to be used as a
pointer to which the other three functional areas of the FDC
can point to. The ID Section is also used by the Formatter
upon design configuration output as an identity header. The
relational model for the Identification Section is depicted
in Figure 10.

The next area of concern is the Design Criteria Section.
The design criteria are also input by the designer at the
initial session with the SDE. The designer responds to
prompts regarding CSDE run parameters and FDC output
metrices. All of the possible design criteria have not yet
been developed and as the CSDE matures additional design

```
                    IDENTIFICATION_TABLE

field    NAME    DATE    PROJECT    ID_PTR    VERSION_NO
-----------------------------------------------------------
type     CHAR    DATE    CHAR       NUMBER    NUMBER
```

Figure 10   Identification Section Model

IDENTIFICATION_TABLE

Field NAME	DATE	PROJECT	ID	STR	ACTUAL_RE
Type	CHA3	DATE	CHA2	9/8/82	N/8925

Figure 19: Identification Table

information may be required. With this in mind, the relational model has been designed for extensibility and flexibility.

The Design Criteria Section covers 2 general areas: 1) CSDE operating instructions and 2) FDC listing criteria. The design criteria for the CSDE operating instructions are very limited at this point and only allow the designer to specify the order in which library volumes and monitor strategies are tested. FDC listing parameters are similarly limited in the present CSDE but as more is studied and known about designer needs, additional information can be gathered from the CSDE process and organized for output. As the system runs now only 2 statistics can be extracted to compare microprocessor feasibility: power consumption and chip count (an approximate cost function). Storage and time comparisons can easily be added to this list. The information is already present in the data base. Additional designer queries will have to added to the frontend and the responsibility of outputting the information will lie with the Formatter. The relational model for the Design Criteria section is shown in Figure 11.

The relational model can be used as follows: 1) the designer specifies the microprocessor and monitor he/she wants to use for each run by placing the order number in the CSDE_RUN field and the respective microprocessor and monitor schemes to be attempted in their respective fields and then

```
field    ID_PTR    CSDE_RUN    MP      MONITOR
------------------------------------------------
type     NUMBER    NUMBER       CHAR    CHAR


field    STORAGE    TIME    POWER    CHIP
------------------------------------------------
type     NUMBER     NUMBER NUMBER   NUMBER
```

Figure 11 Design Criteria Table

2) the designer specifies which design abstractions he/she is interested in for output by responding to SDE system prompts in the corresponding manner. There is no limit to the number of fields in a relational table so as the CSDE grows, numerous designer queries can be placed in the Design Criteria table.

Another important set of information derived from the problem statement input used in the FDC concerns what the present system calls the Application Timing Table (ATT). The ATT information is extracted from the problem statement in CSDL format by the Translator. This information is the set of timing constraints for an application given by the designer. A second set of related information is contained in the current Realization Timing Table (RTT). This information is extracted by the Functional Mapper from the microprocessor realization selected from the library as the mapping takes place. In other words, the RTT contains the possible timing parameters that the microprocessor in question can supply to meet the designer's needs. This information is used by the Timing Analyzer subroutine to compare the timing parameters in each table to test a particular microprocessor for feasibility. If the micro-processor proves feasible, a monitor scheme is then developed by the MONITOR subroutine according to the timing specifications in the RTT.

The original contrants imposed by Matelan and also
used by Ross have been changed for this design study. This
study is testing the concept that the only timing parameter
needed for feasibility testing is the frequency of a con-
tingency. This is represented as rho in the references. If
this concept is proven out in the future, the design and
implementation will have been greatly simplified. The two
tables have also been logically organized into one rela-
tional table. This is so that all the timing related issues
are modularized into one section. A designer will know
where to go to find the answer to any question related to
timing. The translator will still have to enter the
application's timing constraints into the data base and the
Functional Mapper will still have to compute the micropro-
cessor's ability to perform the required operation with
respect to time, but the orientation of the data has been
structured into one logical entity. Figure 12 depicts the
data base structure for the Timing Table entity.

The fourth functional entity in the FDC is the index to
the software and hardware macros needed for each contingency
and task in the design statement. Since each contingency or
task will require more than one primitive to realize its
function, 2 tables are necessary. The first table, as
depicted in Figure 13, consists of the title of the
contingency or task, a primitive and a pointer field. The
pointer field is again used to relate the tables in a many

```
field    CONTINGENCY    TASK    ATT_RHO    TRR_RHO
------------------------------------------------
type     CHAR           CHAR    NUMBER     NUMBER
```

Figure 12   Timing Table

```
                        PRIMARY_INDEX

field     TITLE    PRIMITIVE    POINTER
-------------------------------------------
type      CHAR     CHAR         NUMBER




                      SECONDARY_INDEX

field     POINTER    PRIMITIVE
-----------------------------------
type      NUMBER     CHAR
```

                    Figure 13   Index Tables

to one fashion. The second table is defined by a like pointer field and a field for another primitive of the contingency or task being defined.

The Functional Mapper will retrieve a title line from the RDBT1 (which contains the primitive list and symbol table from the design problem) and place it in the data base. Next, it will search in RDBT2 for matches to each of the primitives in RDBT1 and as these matches are found place them in the FDC. A quick check through the index each time a mapping is found can indicate whether or not a primitive has already been used. This is an important test. For instance, it is not necessary to have redundant hardware or software necessary to implement multiple divide functions.

Once all design primitives have been mapped into a particular design volume the Timing Analyzer will add to the list of primitives any of which are deemed necessary to fulfill the timing requirements. These can all be placed in the FDC under a special title. Upon completion of the microprocessor controller creation, the Formatter will then use the index to indicate which primitives from the microprocessor volume in RDBT2 are to be placed in the final formal output.

The final deisgn configuration is accessed and used by every major subroutine of the CSDE. Care will have to be taken in the implementation of the CSDE application not to overwrite and/or eradicate information. It is recommended

that during the reprogramming of the CSDE to incorporate the
relational data base model, each subroutine be written and
tested separately to ensure that the right information is
going into the right location.

This design presentation has illustrated a conceptual
representation of the data base requirements of the CSDE
through a relational model. The three major aspects of the
CSDE have been incorporated into a logical, cohesive blue-
print. The next chapter illustrates the implementation
methodology using Oracle on the VAX 11/780 with the VMS
operating system as the DBMS.

IV. <u>IMPLEMENTATION</u>

This chapter describes the effort necessary to implement the conceptual design of the CSDE's data base requirements defined in the previous chapter. In addition to the data base implementation, this text also describes the creation of the Application Design Interface (ADI). With this information, the reader will be able to find and use the data base tables, to understand some of the restrictions placed on the implementation by the Oracle DBMS and to have the background necessary to modify the system should it be required by future enhancements.

To implement a data base model, one must first describe that model in terms of the DBM's Data Definition Language (DDL). In this case, the DBMS is Oracle and it uses the standard DDL developed by the designers of system R. [Ref. 6] The format for defining the tables is as follows:

create table <tablename> (<filename><fieldtype>(<field length>));

The DDL format for describing the table in Figure 6 on page 26 is depicted in Figure 14.

Using this example as a reference, specific design features can be pointed out in relationship to Oracle's data types. Three data types were used; CHAR, NUMBER, and DATE.

50

```
create table identification_block (name char(25),
                                   the_date date,
                                   project char(25),
                                   version_no number(6),
                                   id_ptr number(6));
```

Figure 14   DDL Format Example

CHAR allows all characters to be used in the field; upper/lower case and all special characters. This flexibility is required by the need to use mathematical symbols in conjunction with text. Oracle allows CHAR data lengths from 1 to 240. To provide some regularity in the design, CHAR lengths were limited to 80, 25 and 3. In the tables for the data base, 80 and 25 was used for text strings and 3 was used to express mathematical operators. NUMBER is an integer data type and again, to introduce regularity all fields of data type NUMBER were limited to a length of 6; 6 being the maximum length required by any one field. The third and final data type, DATE, is unique in that Oracle prespecifies the length as 7 so it therefore is left blank.

The tables were placed in a file called CREATEDB.UFI under the CSDE subdirectory [.HEATHER] on the VAX 11/780 using the EDT editor, available under the VMS operating system. The listing for the file is available in Appendix A. After the tables were defined in the DDL and put on file, Oracle's User Friendly Interface (UFI) was used to make instantiations of the tables in the DBMS. This is done by:

(1) logging on to Oracle as a CSDE user ty typing Oracle once in VMS.

(2) typing UFI.

(3) answering the screen prompts for user name and password.

(4) typing @CREATEDB.UFI.

52

After a successful table build run, the tables are available through Oracle for data input. The tables remain as defined and do not have to be recreated unless there is a data base design modification.

Data will be input to the tables in three ways: 1) problem statement input via the SDE, 2) creation of the Final Design Configuration by the CSDE's Functional Mapper and 3) creation of the realization libraries through interaction with the ADI. The implementation of the first two categories is beyond the scope of this research. The third category, the creation of a realization library, has been initiated by this thesis effort. An ADI was developed for use by volume designers according to the specifications given in the design chapter. The following is a description of the development of the ADI.

The ADI was developed using Oracle's Application Interface Facility (AIF). The ADI was divided into two parts for reasons of ease of development and ease of use. There is one file that can be used to generate specifications for the hardware primitives and another file to create the software primitives.

To create an ADI one must first invoke Oracle's Interactive Application Generator (IAG). The IAG is a menu driven interpreter that translates the screen designer's input into a CRT screen display for easy volume design input. Each response given by the screen designer to the

IAG's prompts for typing parameters generates a unique sequence of code. In other words, the prompts are not predetermined but vary according to user response. This can be a problem when trying to correct or modify a given screen but a careful study of the Oracle documentation and experimentation do make it possible. To invoke the IAG, one simply logs on to Oracle and then types IAG <filename>. In this case, the filename for hardware is HDESIGN.INP and SDESIGN.INP for software. The IAG will then intiate a series of answer dependent prompts that will define various aspects of the screen interface.

For both the hradware and software screen interfaces, 7 blocks were designed. Each block is represented by a screen. The screens were divided into logically ordered and related entities that also provided a coherent, easy to use presentation. The relational model represented on each screen and the number of times the model is repeated on a single screen is outlined in the following:

(1) IDENTITY_BLOCK   (1)

(2) PARAMETER_BLOCK   (3)

(3) COMMENT_BLOCK   (6)

(4) CALL_BLOCK and INCLUDE_BLOCK   (2 ea.)

(5) ATTRIBUTE_BLOCK and CALC_BLOCK   (2 ea.)

(6) IF_BLOCK   (1)

(7) TEXT_BLOCK   (2)

Each field depicted on the screen is directly related
to a field identified in the data base design. The only
difference between the name of the field on the screen and
in the tables is that the screen name may be slightly more
user friendly than the shortened name used in the data base
design. For instance, MICROPROC in the table IDENTITY_BLOCK,
becomes MICROPROCESSOR on the screen.

In addition to the criteria identified in the data base
tables in Appendix A (data type and length), Oracle's IAG
permits further parameterization of each field represented
in the screen interface. The following is the list of
prompts the IAG uses to qualify various characteristics
about field representation and usage with a short explanation
of each prompt.

(1) Field name: The name given to a field in the data base
    design i.e. MICROPROC.

(2) Length of field/Display length: The length of the
    field identified in the data base design and the length
    of the field to be represented on the screen. In this
    implementation these characteristics were always the
    same.

(3) Is this field in the base table Y/N: A 'Y' or YES
    answer means that the field being described corresponds
    to a column of the table being processed by this block.
    All fields have a YES answer for this question as there
    is a 1:1 correspondence between the tables and the

screen blocks. Exception: The times field in the CALL
and Include Blocks is not depicted on the screen. This
field is input by the CSDE.

(4) Is this field part of the primary key: The fields that
make up the primary key are identified in the tables of
Appendix A by the words 'NOT NULL' next to their data
length parameter. Usually the key is made up of the
pointer and another field within the block that make
the key unique. For example, in the second block,
PARAMETER_BLOCK, the key is made up of the fields
labeled POINTER and PARAMETER.

(5) Field to copy primary key from: Unrelated to this
design, therefore left black by hitting the return key.

(6) Default value: None of the fields have one in this ADI,
therefore left black by hitting the return key.

(7) Page: Identifies which screen page/block the table
is a part of. In this screen design, the hardware
and software files are each made up of 7 pages.

(8) Line: Identifies the line number of a page on which
the field prompt is placed. An effort was made to
place similar or identical fields on the same lines
on the different pages.

(9) Column: Identifies the column number from 1-80 on
which the data will be entered. The prompt will appear
beside the column number identified.

(10) Prompt: Identifies the prompt that appears on the screen, i.e. MICROPROCESSOR.

(11) Display prompt above field Y/N: The answer is NO throughout the screen interface design. A NO answer displays the prompt beside the data entry field display.

(12) Allow field to be entered: All fields are designer enterable except the ID# fields. These fields are generated by a sequel statement embedded in the screen interface design.

(13) Allow fields to be updated: if the answer to the above question was no this prompt does not appear. It also does not appear for fields labeled as a part of the key. Otherwise, all other fields are updateable.

(14) SQL> This is a prompt for a sequel statement related to the field being entered. In this screen design, the SQL statements were used to automatically generate unique ID numbers to be used as part of that tables key. The generic SQL statement is as follows: SELECT MAX (<fieldname + 1>) INTO <fieldname> FROM <tablename>;. The fieldname is the same for both sets of brackets and the tablename is the table in which the fieldname resides. Oracle requires that a dummy record be used to start the process of automatic number generation. An INSERT statement was used to set up a fake record with zeros in all the number data typed

57

fields and 'DUMMY RECORD' in all the character string
fields for each of the tables that used a SQL
statement of this kind. This fact will most likely
have to be considered in the design of the Functional
Mapper. SQL statements can be used for additional
checks for data entry clarification. As more is
known about the requirements of the volume design
effort the SQL statements may become an avenue for
additional data typing or input.

(15) Message if value not found: This and the next prompt
are only generated if a SQL statement has been used.
The message usually generated in this instance relates
that the ID# was not generated. There is no reason to
this author's knowledge why the numbers would not be
generated.

(16) Must value exist: The answer is YES.

(17) Is field fixed length: None of the fields are fixed
length. Fixed length refers to the number of charac-
ters or numbers that must be entered and does not refer
to the maximum length definition.

(18) Auto jump to next field: If any of the fields were
fixed length this could be used. As it is now, the
designer must his <return> to move the cursor to the
beginning of the next field.

(19) Convert field to upper case: This is not necessary in this application.

(20) Help message: This IAG prompt gives the screen interface designer a chance to help the volume designer by clarifying what goes into the field. The current help messages will probably be modified as more is known about the volume design effort.

(21) Lowest Value: None of the fields in this application needed to use this concept.

(22) Highest value: None of the fields in this application needed to use this concept.

Once all the screen criteria have been entered the interface designer must then retype IAG filename for a compile to take place. Appendix 2 is a listing of the IAG compiled code. It is easy to discern from the listing all specific screen design decisions concerning each block and their corresponding fields. Further enlightenment regarding the IAG prompts can be found in the Oracle Manual in the IAF Application Design Reference section.

One Oracle idiosyncracy that should be pointed out to future generations of Oracle users involves the modification process. It is easier to erase an entire field and reanswer the prompts regarding its screen activation than to try to change the individual responses. This is because, as noted before, the prompts are answer dependent.

In general, an effort was made to make the screen interface easy to read, easy to use and regular. This will hopefully make it easy for the volume designer to do his/her job and for future changes to be applied against the screen interface.

V. CONCLUDING REMARKS

The previous pages describe the operations of the CSDE;
the responsibilities of the major subroutines, the data flow
characteristics and the data base requirements. Following
these descriptions, a relational data model is defined and
its implementation on the Oracle DBMS is outlined.

Chapter I was dedicated to introducing the problem.
This was done by describing an analogy to a current techno-
logy and its relationship to the work this thesis effort
supports. Chapter I also laid the foundation for the ensuing
chapters, discribing what was to be discussed in each of them.
Chapter II analyzed and defined the problem; what were the
data base requirements of the CSDE and how did they need to
be supported. Chapter III described in detail the rela-
tional model that can support those requirements. Following
this, Chapter IV described the step by step procedures
needed to implement the relational model on a general
purpose DBMS - Oracle.

Throughout this thesis effort an attempt was made to
recommend possible uses of the data base by the CSDE
subroutines. The final incorporation of the relational
model into the CSDE's subroutines will depend on individual
expertise and experience. However, there are many things

to be considered in this future implementation and in the following paragraphs I will briefly highlight some of these.

The next obvious step is to use the relational model in the CSDE's subroutines and make use of the Oracle DBMS. The first part of this step would be to design and implement a Translator based on the relational model. As was previously noted, the purpose of the Translator subroutine is to make the SDE input information available to the CREATEDB1 module, which then places the data in the Primitive List and Symbol Table tables. In parallel to the performance of this task, a microprocessor design volume will also have to be placed in the data base tables. This can be done using the Application Design Interface (ADI) for those designs currently being developed or through an interface translator for those preexisting microprocessor designs, where it would be too cumbersome to reenter the data by hand. Onec these entities are in place, the Functional Mapper can be designed.

Relational DBMS's are a relatively new technology. Their use in CAD systems is even more recent. Hence, the amount of applicable literature is limited. However, one thing the literature does reflect is the fact that relational data bases are generally slow. [Ref. 7]

In light of this fact, the designer of any one of the subroutines using the relational model will have to determine query algorithms that optimize Oracle's efficiency and

effectiveness. There are characteristics of the CSDE's use of the data base that will weight the construction of such algorithms a certain way. For instance, when CREATEDB1 enters data into the Primitive List and Symbol Table it will do so sequentially. There is no random generation of primitives and their arguments. A Title group is defined with the macros that will perform the given function and placed in the Primitive List. The arguments for each macro are then supplied with their corresponding precision parameters sequentially.

Therefore, there will be no need for the Oracle system to insert records between existing records. All records will be added to the end of the last record in the table. The algorithm should be designed to take advantage of this fact if at all possible.

The query algorithms for the Timing Analyzer and the Formatter will be variations on this select, compare and insert theme. This variation will enable the programmer to design, test and conduct performance evaluation on the various related algorithms using the Oracle DBMS.

If during the creation of the query algorithms it becomes necessary to modify the data base in order to optimize Oracle's effectiveness, this can be easily done. This is one of the benefits of a relational data base model. The fact that the functional capabilities are incorporated in a single module make it an easy base to transform.

This is in effect the major contribution of this research. Through this effort the CSDE's data requirements were thoroughly analyzed and the information was encapsulated into localized entities which form the basic buidling blocks of the 3 major complex data structures: 1) problem statement input, 2) microprocessor volume library 2) microprocessor volume library and 3) the final design configuration. These three structures, because of their final relational form, can be easily modified. However, the identification and cataloguing of their purpose and function will probably not change drastically. The changes that will come will probably be additions to the data base as new requirements become apparent and necessary.

From here then, the next step is to verify that this plan can be used effectively by the Oracle DBMS. If it is discovered that Oracle can not meet the requirements of the CSDE project, this thesis has provided a sound analysis of the data base requirements, defined in Chapter II and realized into relational form in Chapter III, for implementation on any future system.

A User's Guide for the ADI is on file with the CSDE project.

APPENDIX A
CREATEDB UFI TABLES

```
CREATE TABLE PRIMITIVELIST   (TITLE CHAR(25) NOT NULL,
                              MACRONAME CHAR(25) NOT NULL,
                              POINTERX NUMBER(6));
!
!
! NOT NULL INDICATES FIELD IS A PART OF THE KEY
!
!
CREATE TABLE SYMBOLTABLE   (POINTERX NUMBER(6) NOT NULL,
                            ARGUMENT CHAR(25) NOT NULL,
                            MINIMUM NUMBER(6),
                            MAXIMUM NUMBER(6));
!
!
! NEXT 9 BLOCKS MAKE UP THE HARDWAREMACRO TABLE
! FOR THE REALIZATION VOLUME DESIGN
!
!
CREATE TABLE IDENTITYBLOCK   ( MICROPROC CHAR(25) NOT NULL,
                              HWMACRO CHAR(25) NOT NULL,
                              MILLIWATTS NUMBER(6),
                              CHIPCOUNT NUMBER(6),
                              LATENCY NUMBER(6),
                              POINTER NUMBER(6));
!
!
CREATE TABLE PARAMETERBLOCK ( POINTER NUMBER(6) NOT NULL,
                              PARAMETER CHAR(25) NOT NULL,
                              MINIMUM NUMBER(6),
                              MAXIMUM NUMBER(6));
!
!
CREATE TABLE COMMENTBLOCK ( POINTER NUMBER(6) NOT NULL,
                            COMMENT CHAR(80),
                            COMID NUMBER(6) NOT NULL);
!
!
CREATE TABLE INCLUDEBLOCK ( POINTER NUMBER(6) NOT NULL,
                            INCLUDE CHAR(25),
                            INCID NUMBER(6) NOT NULL,
                            TIMES NUMBER(6));
!
!
CREATE TABLE CALLBLOCK ( POINTER NUMBER(6) NOT NULL,
                         CALL CHAR(25),
                         CALLID NUMBER(6) NOT NULL,
                         TIMES NUMBER(6));
!
!
```

```
CREATE TABLE CALCBLOCK ( POINTER NUMBER(6) NOT NULL,
                         GLOBVAR CHAR(25),
                         OP CHAR(3),
                         VAR CHAR(25),
                         RSLT CHAR(25),
                         CALCID NUMBER(6) NOT NULL);
:
:
CREATE TABLE ATTRIBUTEBLOCK ( POINTER NUMBER(6) NOT NULL,
                              LOCALVAR CHAR(25),
                              OP CHAR(3),
                              VAR CHAR(25),
                              RSLT CHAR(25),
                              ATTRID NUMBER(6) NOT NULL);
:
:
CREATE TABLE IFBLOCK ( POINTER NUMBER(6) NOT NULL,
                       VARIABLE1 CHAR(25),
                       OP CHAR(3),
                       VARIABLE2 CHAR(25),
                       IFMP CHAR(25),
                       IFMACRO CHAR(25),
                       IFREQ CHAR(25),
                       ID NUMBER(6),
                       IFID NUMBER(6) NOT NULL);
:
:
CREATE TABLE TEXTBLOCK ( POINTER NUMBER(6) NOT NULL,
                         TEXT CHAR(80),
                         TEXTID NUMBER(6) NOT NULL);
:
:
: THE NEXT 9 BLOCKS CONTAIN THE TABLES NECESSARY FOR THE
: SOFTWARE PRIMITIVES FOR THE REALIZATION VOLUME
:
:
CREATE TABLE SIDENTITYBLOCK  ( MICROPROC CHAR(25) NOT NULL,
                              SWMACRO CHAR(25) NOT NULL,
                              STORAGE NUMBER(6),
                              TIME NUMBER(6),
                              EXTREF NUMBER(6),
                              POINTER NUMBER(6));
:
:
CREATE TABLE SPARAMETERBLOCK ( POINTER NUMBER(6) NOT NULL,
                              PARAMETER CHAR(25) NOT
NULL,
                              MINIMUM NUMBER(6),
                              MAXIMUM NUMBER(6));
```

66

```
!
!
CREATE TABLE SCOMMENTBLOCK ( POINTER NUMBER(6) NOT NULL,
                             COMMENT CHAR(80),
                             COMID NUMBER(6) NOT NULL);
!
!
CREATE TABLE SINCLUDEBLOCK ( POINTER NUMBER(6) NOT NULL,
                             INCLUDE CHAR(25),
                             INCID NUMBER(6) NOT NULL,
                             TIMES NUMBER(6));
!
!
CREATE TABLE SCALLBLOCK ( POINTER NUMBER(6) NOT NULL,
                          CALL CHAR(25),
                          CALLID NUMBER(6) NOT NULL,
                          TIMES NUMBER(6));
!
!
CREATE TABLE SCALCBLOCK ( POINTER NUMBER(6) NOT NULL,
                          GLOBVAR CHAR(25),
                          OP CHAR(3),
                          VAR CHAR(25),
                          RSLT CHAR(25),
                          CALCID NUMBER(6) NOT NULL);
!
!
CREATE TABLE SATTRIBUTEBLOCK ( POINTER NUMBER(6) NOT NULL,
                               LOCALVAR CHAR(25),
                               OP CHAR(3),
                               VAR CHAR(25),
                               RSLT CHAR(25),
                               ATTRID NUMBER(6) NOT NULL);
!
!
CREATE TABLE SIFBLOCK ( POINTER NUMBER(6) NOT NULL,
                        VARIABLE1 CHAR(25),
                        OP CHAR(3),
                        VARIABLE2 CHAR(25),
                        IFMP CHAR(25),
                        IFMACRO CHAR(25),
                        IFREQ CHAR(25),
                        ID NUMBER(6),
                        IFID NUMBER(6) NOT NULL);
!
!
CREATE TABLE STEXTBLOCK ( POINTER NUMBER(6) NOT NULL,
                          TEXT CHAR(80),
                          TEXTID NUMBER(6) NOT NULL);
```

67

```
:
:
: THE NEXT TABLE REPRESENTS THE RELATIONAL MODEL FOR THE
: IDENTIFICATION TABLE
:
:
CREATE TABLE IDENTIFICATION ( NAME CHAR(25) NOT NULL,
                              THEDATE DATE NOT NULL,
                              PROJECT CHAR(25) NOT NULL,
                              VERSIONNO NUMBER(6) NOT NULL,
                              IDPTR NUMBER(6) NOT NULL);
:
:
: THE NEXT TABLE REPRESENTS THE DESIGN CRITERIA TABLE
:
:
CREATE TABLE DESIGNCRITERIA ( IDPTR NUMBER(6) NOT NULL,
                             CSDERUN NUMBER(3) NOT NULL,
                             MP CHAR(25),
                             MONITOR CHAR(25),
                             STORAGE NUMBER(6),
                             TIME NUMBER(6),
                             POWER NUMBER(6),
                             CHIP NUMBER(6));
:
:
: THE FOLLOWING TABLE REPRESENTS THE TIMING TABLE
:
:
CREATE TABLE TIMING ( CONTINGENCY CHAR(25) NOT NULL,
                      TASK CHAR(25) NOT NULL,
                      ATTRHO NUMBER(6),
                      RTTRHO NUMBER(6));
:
:
: THE NEXT TWO TABLES REPRESENT THE RELATIONAL MODEL FOR THE
: HARDWARE AND SOFTWARE PRIMITIVE INDEX
:
:
CREATE TABLE PRIMARYINDEX ( TITLE CHAR(25) NOT NULL,
                           PRIMITIVE CHAR(25) NOT NULL,
                           PRIMPTR NUMBER(6));
:
:
CREATE TABLE SECONDARYINDEX ( PRIMPTR NUMBER(6) NOT NULL,
                             PRIMITIVE CHAR(25) NOT NULL);
:
```

68

APPENDIX B

INTERACTIVE APPLICATION GENERATOR LISTING

```
;Database :
CREATEDB.UFI
;ORACLE workspace size / Context areas :

;Block name / Description :
SOFTIDENTITY/SOFTWARE PERFORMANCE SPECS
;Table name :
SIDENTITYBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
1
;Field name :
MICROPROC
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
1
;Line :
2
;Column :
16
;Prompt :
MICROPROCESSOR
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER MICROPROCESSOR TYPE i.e. 8080
;Lowest value :
```

```
;Highest value :

;Field name :
SWMACRO
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
1
;Line :
4
;Column :
20
;Prompt :
SOFTWARE PRIMITIVE
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER SOFTWARE PRIMTIVE TO BE DESCRIBED
;Lowest value :

;Highest value :

;Field name :
STORAGE
;Type of field :
NUMBER
;Length of field / Display length :
6/6
```

```
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
1
;Line :
6
;Column :
9
;Prompt :
STORAGE
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
HOW MANY BYTES IS THE PRIMITIVE CODE
;Lowest value :

;Highest value :

;Field name :
TIME
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
```

71

```
1
;Line :
8
;Column :
6
;Prompt :
TIME
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
HOW MANY CLOCK (T) STATES DOES THIS PRIMITIVE REQUIRE?
;Lowest value :

;Highest value :

;Field name :
EXTREF
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
1
;Line :
10
;Column :
21
;Prompt :
EXTERNAL REFERENCES
```

```
;Display promot above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
NUMBER OF MEMORY STATES
;Lowest value :

;Highest value :

;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :
99
;Page :
1
;Line :
12
;Column :
9
;Promot :
POINTER
;Display promot above field Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX(POINTER + 1)
INTO POINTER
```

FROM SIDENTITYBLOCK

;Message if value not found :
POINTER VALUE NOT GENERATED BY DBMS
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
PARAMETERS/PRIMITIVE'S I/O VARIABLES
;Table name :
SPARAMETERBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
6
;Base crt line ?
3
;How many physical lines per record ?
3
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
2
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

74

```
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED IN LAST BLOCK
;Lowest value :

;Highest value :

;Field name :
PARAMETER
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
2
;Line :
1
;Column :
40
;Prompt :
PARAMETER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
```

```
;Help message :
ENTER INPUT OR OUTPUT PARAMETERS FOR PRIMITIVE
;Lowest value :

;Highest value :

;Field name :
MINIMUM
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
2
;Line :
2
;Column :
9
;Prompt :
MINIMUM
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER MINIMUM PRECISION FOR ARGUMENTS i.e. 8 bits
;Lowest value :

;Highest value :
```

76

```
;Field name :
MAXIMUM
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
2
;Line :
2
;Column :
40
;Prompt :
MAXIMUM
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER MAXIMUM PRECISIONFOR ARGUMENT i.e. 16 bits
;Lowest value :

;Highest value :

;Field name :

;Block name / Description :
COMMENT/COMMENTS FOR PRIMITIVE
```

```
;Table name :
SCOMMENTBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
5
;Base crt line ?
4
;How many ohysical lines per record ?
4
;Field name :
POINTER
;Tyoe of field :
NUMBER
;Length of field / Disolay length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field oart of the orimary key Y/N :
Y
;Field to copy orimary key from :

;Default value :

;Paoe :
3
;Line :
1
;Column :
9
;Promot :
POINTER
;Disolay oromot aoove field Y/N :
N
;Display oromot once for olock Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed lenath Y/N :
N
;Auto jumo to next field Y/N :
N
;Convert field to uooer case Y/N :
N
;Helo message :
ENTER POINTER VALUE GENERATED IN FIRST BLOCK
;Lowest value :
```

```
;Highest value :

;Field name :
COMMENT
;Type of field :
CHAR
;Length of field / Display length :
80/80
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
3
;Line :
2
;Column :
9
;Prompt :
COMMENT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
COMMENTS WILL BE IN THE ORDER OF INPUT
;Lowest value :

;Highest value :

;Field name :
COMID
```

```
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
3
;Line :
3
;Column :
50
;Prompt :
ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX( COMID + 1)
INTO COMID
FROM SCOMMENTBLOCK

;Message if value not found :
ID  NOT GENERATED BY DBMS
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
INCLUDE/ INCLUDE DESCRIPTIONS
;Table name :
INCLUDEBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
;Base crt line ?
2
;How many physical lines per record ?
4
```

```
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
4
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED ON FIRST BLOCK
;Lowest value :

;Highest value :

;Field name :
INCLUDE
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
```

```
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
4
;Line :
1
;Column :
40
;Prompt :
INCLUDE
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER NAME OF H/W OR S/W PRIMITIVE TO BE INCLUDED
;Lowest value :

;Highest value :

;Field name :
INCID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :
```

82

```
;Default value :
99
;Page :
4
;Line :
2
;Column :
12
;Promot :
INCLUDE ID
;Display oromot above field Y/N :
N
;Display oromot once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (INCID + 1)
INTO INCID
FROM INCLUDEBLOCK

;Message if value not found :
ID  NOT GENERATED BY DBMS
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
CALL BLOCK/ MACRO TO BE USED BY CURRENT PRIMITIVE
;Table name :
SCALLBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
;Base crt line ?
14
;How many ohysical lines oer record ?
4
;Field name :
POINTER
;Tyoe of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base taole Y/N :
Y
;Is this field oart of the orimary key Y/N :
Y
```

83

```
;Field to copy primary key from :

;Default value :

;Page :
4
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER VALUE GENERATED ON FIRST PAGE
;Lowest value :

;Highest value :

;Field name :
CALL
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
4
;Line :
1
;Column :
```

```
40
;Promot :
CALL
;Display promot above field Y/N :
N
;Display promot once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER H/W OR S/W PRIMITIVE TO BE CALLED FROM CURRENT MACRO
;Lowest value :

;Highest value :

;Field name :
CALLID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
4
;Line :
2
;Column :
10
;Promot :
CALL ID
```

```
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX ( CALLID + 1 )
INTO CALLID
FROM SCALLBLOCK

;Message if value not found :
ID  NOT GENERATED!!
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
CALC/ CALCULATION OF GLOBAL VARIABLES
;Table name :
CALCBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
;Base crt line ?
2
;How many physical lines per record ?
5
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
5
;Line :
1
;Column :
9
```

```
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED ON FIRST PAGE
;Lowest value :

;Highest value :

;Field name :
GLOBVAR
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
1
;Column :
40
;Prompt :
GLOBAL VARIABLE
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
```

```
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER GLOBAL VARIABLE TO BE CALCULATED
;Lowest value :

;Highest value :

;Field name :
OP
;Type of field :
CHAR
;Length of field / Display length :
3/3
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
2
;Column :
10
;Prompt :
OPERATOR
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
```

```
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER OPERATOR **,*,+,-,/
;Lowest value :

;Highest value :

;Field name :
VAR
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
2
;Column :
40
;Prompt :
VARIABLE
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
```

```
N
;Help message :
GLOBAL VARIABLE (OPERATOR) VARIABLE = RESULT
;Lowest value :

;Highest value :

;Field name :
RSLT
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
3
;Column :
40
;Prompt :
RESULT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
GLOBAL VARIABLE (OPERATOR) VARIABLE = RESULT
;Lowest value :
```

```
;Highest value :

;Field name :
CALCID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
5
;Line :
4
;Column :
10
;Prompt :
CALC ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (CALCID + 1 )
INTO CALCID
FROM CALCBLOCK

;Message if value not found :
ID  NOT GENERATED BY THE SYSTEM
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
ATTRIBUTE
;Table name :
ATTRIBUTEBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
```

```
;Base crt line ?
14
;How many pnysical lines per record ?
5
;Field name :
POINTER
;Type of field :
NUMBER
;Lenath of field / Display lenath :
6/6
;Is this field in the base taple Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
5
;Line :
1
;Column :
9
;Promot :
POINTER
;Display promot above field Y/N :
N
;Display promot once for plock Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed lenath Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER VALUE GENERATED ON FIRST PAGE
;Lowest value :

;Highest value :

;Field name :
LOCALVAR
;Type of field :
```

```
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
1
;Column :
40
;Prompt :
LOCAL VARIABLE
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER LOCAL VARIABLE WHOSE VALUE IS TO BE CALCULATED
;Lowest value :

;Highest value :

;Field name :
OP
;Type of field :
CHAR
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
```

```
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
2
;Column :
10
;Prompt :
OPERATOR
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER **,*,+,-, OR /
;Lowest value :

;Highest value :

;Field name :
VAR
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
```

```
5
;Line :
2
;Column :
40
;Promot :
VARIABLE
;Display promot above field Y/N :
N
;Display promot once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
LOCAL VARIABLE (OPERATOR) VARIABLE = RESULT
;Lowest value :

;Highest value :

;Field name :
RSLT
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
3
;Column :
40
```

```
;Promot :
RESULT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
LOCAL VARIABLE (OPERATOR) VARIABLE = RESULT
;Lowest value :

;Highest value :

;Field name :
ATTRID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
5
;Line :
4
;Column :
15
;Promot :
ATTRIBUTE ID
;Display prompt above field Y/N :
```

N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (ATTRID + 1)
INTO ATTRID
FROM SATTRIBUTEBLOCK

;Message if value not found :
ID NOT GENERATED BY SYSTEM
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
IF/SOFTWARE MACRO IF STATEMENTS
;Table name :
SIFBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
1
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
6
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :

```
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER VALUE GENERATED ON FIRST PAGE
;Lowest value :

;Highest value :

;Field name :
VARIABLE1
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
1
;Column :
40
;Prompt :
VARIABLE 1
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
```

98

```
;Convert field to upper case Y/N :
N
;Help message :
VARIABLE 1 (OP) VARIABLE 2 = RESULT
;Lowest value :

;Highest value :

;Field name :
OP
;Type of field :
CHAR
;Length of field / Display length :
3/3
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
10
;Prompt :
OPERATOR
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :

;Lowest value :

;Highest value :
```

```
;Field name :
VARIABLE2
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
40
;Prompt :
VARIABLE 2
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
VARIABLE 1 (OP) VARIABLE 2 = RESULT
;Lowest value :

;Highest value :

;Field name :
IFMP
;Type of field :
CHAR
;Length of field / Display length :
25/25
```

100

```
;Is this field in the base table Y/N :
Y
;Is this field part of the orimary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
17
;Promot :
MICROPROCESSOR
;Display oromot above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jumo to next field Y/N :
N
;Convert field to uoper case Y/N :
N
;Helo message :
ENTER MICROPROCESSOR TYPE i.e. 8080
;Lowest value :

;Highest value :

;Field name :
IFMACRO
;Type of field :
CHAR
;Length of field / Disolay length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the orimary key Y/N :
N
;Default value :

;Page :
```

```
6
;Line :
6
;Column :
27
;Prompt :
S/W OR H/W PRIMITIVE NAME
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
NAME OF MACRO IF STATEMENT APPLIES TO
;Lowest value :

;Highest value :

;Field name :
IFREQ
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
8
;Column :
16
;Prompt :
TIMES REQUIRED
```

102

```
;Display promot above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
NUMBER OF TIMES PRIMITIVE WILL BE USED
;Lowest value :

;Highest value :

;Field name :
IFID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
6
;Line :
10
;Column :
30
;Promot :
IF ID
;Display promot above field Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
```

```
SELECT MAX ( IFID + 1)
INTO IFID
FROM SIFBLOCK

;Message if value not found :
ID  NOT GENERATED BY SYSTEM
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
TEXT/MACRO ASSEMBLY PROGRAM
;Table name :
STEXTBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
5
;Base crt line ?
4
;How many physical lines per record ?
4
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
7
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
```

```
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER VALUE GENERATED ON FIRST PAGE
;Lowest value :

;Highest value :

;Field name :
TEXT
;Type of field :
CHAR
;Length of field / Display length :
80/80
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
7
;Line :
2
;Column :
6
;Prompt :
TEXT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
```

105

```
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER ASSEMBLY CODE
;Lowest value :

;Highest value :

;Field name :
TEXTID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
7
;Line :
4
;Column :
10
;Prompt :
TEXT ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (TEXTID + 1)
INTO TEXTID
FROM STEXTBLOCK

;Message if value not found :
ID   NOT GENERATED BY SYSTEM
;Must value exist Y/N :
Y
;Field name :
```

```
;Block name / Description :
%END
```

```
;Database :
CREATEDB.UFI
;ORACLE workspace size / Context areas :

;Block name / Description :
IDENTITY/HARDWARE PERFORMANCE SPECIFICATIONS
;Table name :
IDENTITYBLOCK
;Check for uniqueness before inserting Y/N :
Y
;Display/Buffer how many records :
1
;Field name :
MICROPROC
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
1
;Line :
2
;Column :
16
;Prompt :
MICROPROCESSOR
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER MICROPROCESSOR TO BE DESCRIBED
```

108

```
;Lowest value :

;Highest value :

;Field name :
HWMACRO
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
1
;Line :
4
;Column :
20
;Prompt :
HARDWARE PRIMITIVE
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER NAME OF H/W MACRO TO BE DESCRIBED i.e. H.RAM256
;Lowest value :

;Highest value :

;Field name :
MILLIWATTS
;Type of field :
NUMBER
```

```
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
1
;Line :
6
;Column :
12
;Prompt :
MILLIWATTS
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POWER CONSUMPTION REQUIREMENTS FOR MACRO
;Lowest value :

;Highest value :

;Field name :
CHIPCOUNT
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
```

```
;Default value :

;Page :
1
;Line :
8
;Column :
12
;Promot :
CHIP COUNT
;Display promot above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER NUMBER OF CHIP COMPONENTS REQUIRED BY MACRO = APPR.
COST
;Lowest value :

;Highest value :

;Field name :
LATENCY
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
1
;Line :
```

111

```
10
;Column :
9
;Prompt :
LATENCY
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :

;Lowest value :

;Highest value :

;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :
99
;Page :
1
;Line :
12
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
```

```
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (POINTER +1)
INTO POINTER
FROM IDENTITYBLOCK

;Message if value not found :
POINTER NOT GENERATED BY SYSTEM!!!!
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
PARAMETER/CONTAINS MACRO'S ARGUMENTS WITH PRECISION REQ'TS
;Table name :
PARAMETERBLOCK
;Check for uniqueness before inserting Y/N :
Y
;Display/Buffer how many records :
6
;Base crt line ?
3
;How many physical lines per record ?
3
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
2
;Line :
1
;Column :
9
;Prompt :
POINTER
```

```
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED IN PREVIOUS BLOCK
;Lowest value :

;Highest value :

;Field name :
PARAMETER
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
2
;Line :
1
;Column :
40
;Prompt :
PARAMETER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
```

```
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER INPUT OR OUTPUT PARAMETER FOR PRIMITIVE
;Lowest value :

;Highest value :

;Field name :
MINIMUM
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
2
;Line :
2
;Column :
9
;Prompt :
MINIMUM
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
```

115

```
;Auto jumo to next field Y/N :
N
;Convert field to upoer case Y/N :
N
;Helc message :
ENTER MINIMJM PRECISION FOR PARAMETER i.e. 8 bits
;Lowest value :

;Highest value :

;Field name :
MAXIMUM
;Type of field :
NUMBER
;Length of field / Disolay lenath :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the orimary kev Y/N :
N
;Default value :

;Page :
2
;Line :
2
;Column :
40
;Promot :
MAXIMUM
;Disolay oromot above field Y/N :
N
;Disolay oromot once for nlock Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to he uodated Y/N :
Y
;SQL>

;Is field mandatorv Y/N :
N
;Is field fixed lenath Y/N :
N
;Auto jumo to next field Y/N :
N
;Convert field to upoer case Y/N :
N
```

116

```
;Help message :
ENTER MAXIMUM BIT PRECISION FOR PARAMETER i.e. 16 bits
;Lowest value :

;Highest value :

;Field name :

;Block name / Description :
COMMENTS/COMMENTS FOR PRIMITIVE
;Table name :
COMMENTBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
5
;Base crt line ?
4
;How many physical lines per record ?
4
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
3
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
```

```
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED IN FIRST BLOCK OF PRIMITIVE
DESCRIPTION
;Lowest value :

;Highest value :

;Field name :
COMMENT
;Type of field :
CHAR
;Length of field / Display length :
80/80
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
3
;Line :
2
;Column :
9
;Prompt :
COMMENT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
```

118

```
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
COMMENT PRIMITIVE FOR BETTER UNDERSTANDING OF DESIGN
;Lowest value :

;Highest value :

;Field name :
COMID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
3
;Line :
3
;Column :
50
;Prompt :
COMMENT ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (COMID + 1)
INTO COMID
FROM COMMENTBLOCK

;Message if value not found :
COMMENT ID  NOT GENERATED!!!
;Must value exist Y/N :
Y
```

```
;Field name :

;Block name / Description :
INCLUDE/ DESCRIPTION OF PRIMITIVE TO BE INCLUDED WITH CURRENT
PRIMITIVE
;Table name :
INCLUDEBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
;Base crt line ?
2
;How many physical lines per record ?
4
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
4
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
```

120

N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED FROM FIRST BLOCK/PAGE
;Lowest value :

;Highest value :

;Field name :
INCLUDE
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
4
;Line :
1
;Column :
40
;Prompt :
INCLUDE
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :

```
ENTER HARDWARE OR SOFTWARE PRIMITIVE TO BE INCLUDED
;Lowest value :

;Highest value :

;Field name :
INCID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
4
;Line :
2
;Column :
12
;Prompt :
INCLUDE ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (INCID + 1)
INTO INCID
FROM INCLUDEBLOCK

;Message if value not found :
INCLUDE ID  NOT GENERATED BY SYSTEM!!!!
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
CALL/DESCRIPTION OF PRIMITIVE TO BE CALLED BY
CURRENTPRIMITIVE
;Table name :
```

```
CALLBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
;Base crt line ?
14
;How many physical lines per record ?
4
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
4
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED IN FIRST BLOCK
;Lowest value :
```

123

```
;Highest value :

;Field name :
CALL
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
4
;Line :
1
;Column :
40
;Prompt :
CALL
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER H/W OR S/W PRIMITIVE TO BE CALLED FROM CURRENT MACRO
;Lowest value :

;Highest value :

;Field name :
```

```
CALLID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
4
;Line :
2
;Column :
10
;Prompt :
CALL ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :

;SQL>
SELECT MAX (CALLID + 1)
INTO CALLID
FROM CALLBLOCK

;Message if value not found :
CALL ID  NOT GENERATED BY SYSTEM!!!!
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
CALC/CALCULATION OF GLOBAL VARIABLES
;Table name :
CALCBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
;Base crt line ?
2
```

```
;How many physical lines per record ?
5
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
5
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED IN FIRST BLOCK
;Lowest value :

;Highest value :

;Field name :
GLOBVAR
;Type of field :
CHAR
```

```
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
1
;Column :
40
;Prompt :
GLOBAL VARIABLE
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER GLOBAL VARIABLE TO BE CALCULATED
;Lowest value :

;Highest value :

;Field name :
OP
;Type of field :
CHAR
;Length of field / Display length :
3/3
;Is this field in the base table Y/N :
Y
```

127

```
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
2
;Column :
10
;Prompt :
OPERATOR
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER OPERATOR **,*,+,-,/
;Lowest value :

;Highest value :

;Field name :
VAR
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :
```

```
;Page :
5
;Line :
2
;Column :
40
;Prompt :
VARIABLE
;Display promot above field Y/N :
N
;Display promot once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
GLOBAL VARIABLE (OPERATOR) VARIABLE = RSLT
;Lowest value :

;Highest value :

;Field name :
RSLT
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
3
```

129

```
;Column :
40
;Prompt :
RESULT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
GLOBAL VARIABLE (OPERATOR) VARIABLE = RESULT
;Lowest value :

;Highest value :

;Field name :
CALCID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
5
;Line :
4
;Column :
10
```

```
;Prompt :
CALC ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>
SELECT MAX (CALCID + 1)
INTO CALCID
FROM CALCBLOCK

;Message if value not found :
CALC ID  NOT GENERATED BY SYSTEM!!!!
;Must value exist Y/N :
Y
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
VALUE SHOULD BE GENERATED BY SYSTEM AT COMMIT TIME
;Lowest value :

;Highest value :

;Field name :

;Block name / Description :
ATTRIBUTE/ CALCULATION OF GLOBAL VARIABLES
;Table name :
ATTRIBUTEBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
2
;Base crt line ?
14
;How many physical lines per record ?
5
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
```

```
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
5
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED FROM FIRST BLOCK/PAGE
;Lowest value :

;Highest value :

;Field name :
LOCALVAR
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :
```

```
;Page :
5
;Line :
1
;Column :
40
;Promot :
LOCAL VARIABLE
;Display promot above field Y/N :
N
;Display promot once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER LOCAL VARIABLE VALUE TO BE CALCULATED
;Lowest value :

;Highest value :

;Field name :
OP
;Type of field :
CHAR
;Length of field / Display length :
3/3
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
```

```
2
;Column :
10
;Promot :
OPERATOR
;Display promot above field Y/N :
N
;Display promot once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
LOCAL VARIABLE (**,*,+,-,/) VARIABLE = RESULT
;Lowest value :

;Highest value :

;Field name :
VAR
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
2
;Column :
40
;Promot :
```

```
VARIABLE
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
LOCAL VARIABLE (OPERATOR) VARIABLE = RESULT
;Lowest value :

;Highest value :

;Field name :
RSLT
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
5
;Line :
3
;Column :
40
;Prompt :
RESULT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
```

```
N
;Allow field to be entered Y/N :
Y
;Allow field to be uodated Y/N :
Y
;SQL>

;Is field mandatorv Y/N :
NN
;Is field fixed length Y/N :
N
;Auto jumo to next field Y/N :
N
;Convert field to uoper case Y/V :
N
;Helo message :
RESULT CAN CONTAIN INTERMEDIATE OR FINAL RESULTS OF VAR
CALC.
;Lowest value :

;Highest value :

;Field name :
ATTRID
;Tvoe of field :
NUMBER
;Length of field / Disolay lenqth :
6/6
;Is this field in the base table Y/N :
Y
;Is this field oart of the orimary kev Y/N :
Y
;Field to cooy orimary key from :

;Default value :
99
;Page :
5
;Line :
4
;Column :
15
;Promot :
ATTRIBUTE ID
;Disolay oromot above field Y/N :
N
;Disolav oromot once for block Y/N :
N
```

```
;Allow field to be entered Y/N :
Y
;SQL>
SELECT MAX (ATTRID + 1)
INTO ATTRID
FROM ATTRIBUTEBLOCK

;Message if value not found :
SYSTEM DID NOT GENERATE ATTRIBUTE ID
;Must value exist Y/N :
Y
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ATTRIBUTE ID IS SYSTEM GENERATED AND IS A KEY FIELD
;Lowest value :

;Highest value :

;Field name :

;Block name / Description :
IF/DESCRIBES PRIMTIVE IF STATEMENTS
;Table name :
IFBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
1
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :

;Page :
```

```
6
;Line :
1
;Column :
9
;Promot :
POINTER
;Display oromot above field Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed lenath Y/N :
N
;Auto jumo to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Helo message :
ENTER POINTER VALUE GENERATED IN FIRST BLOCK/PAGE
;Lowest value :

;Highest value :

;Field name :
VARIABLE1
;Type of field :
CHAR
;Lenath of field / Disolav length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field oart of the orimary key Y/N :
N
;Default value :

;Page :
6
;Line :
1
;Column :
40
;Promot :
VARIABLE1
;Display oromot above field Y/N :
N
;Allow field to be entered Y/N :
```

```
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
VARIABLE1 (OP) VARIABLE2 = RESULT
;Lowest value :

;Highest value :

;Field name :
OP
;Type of field :
CHAR
;Length of field / Display length :
3/3
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
10
;Prompt :
OPERATOR
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
```

139

```
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
RELATIONAL OPERATOR (=,<,>,<=,>=)
;Lowest value :

;Highest value :

;Field name :
VARIABLE2
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
2
;Column :
40
;Prompt :
VARIABLE2
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
```

```
N
;Help message :
VARIABLE1 (REL OP) VARIABLE2 = RESULT
;Lowest value :

;Highest value :

;Field name :
IFMP
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
4
;Column :
16
;Prompt :
MICROPROCESSOR
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER MICROPROCESSOR TYPE i.e. 8080
;Lowest value :

;Highest value :
```

```
;Field name :
IFMACRO
;Type of field :
CHAR
;Length of field / Display length :
25/25
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
6
;Column :
27
;Prompt :
S/W OR H/W PRIMITIVE NAME
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER MACRO TO BE CALLED OR INCLUDED
;Lowest value :

;Highest value :

;Field name :
IFREQ
;Type of field :
NUMBER
;Length of field / Display length :
```

```
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
6
;Line :
8
;Column :
16
;Prompt :
TIMES REQUIRED
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER NUMBER OF TIMES PRIMTIVE WILL BE USED
;Lowest value :

;Highest value :

;Field name :
IFID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :
```

143

```
;Default value :
99
;Page :
6
;Line :
10
;Column :
30
;Prompt :
IF ID
;Display prompt above field Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (IFID + 1)
INTO IFID
FROM IFBLOCK

;Message if value not found :
ID  NOT GENERATED
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :
TEXT/ MACRO ASSEMBLY PROGRAM
;Table name :
TEXTBLOCK
;Check for uniqueness before inserting Y/N :
N
;Display/Buffer how many records :
5
;Base crt line ?
3
;How many physical lines per record ?
4
;Field name :
POINTER
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
```

```
;Field to copy primary key from :

;Default value :

;Page :
7
;Line :
1
;Column :
9
;Prompt :
POINTER
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;SQL>

;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER POINTER VALUE GENERATED IN THE FIRST BLOCK
;Lowest value :

;Highest value :

;Field name :
TEXT
;Type of field :
CHAR
;Length of field / Display length :
80/80
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
N
;Default value :

;Page :
7
;Line :
2
```

```
;Column :
6
;Prompt :
TEXT
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
Y
;Allow field to be updated Y/N :
Y
;SQL>

;Is field mandatory Y/N :
N
;Is field fixed length Y/N :
N
;Auto jump to next field Y/N :
N
;Convert field to upper case Y/N :
N
;Help message :
ENTER ASSEMBLY CODE LINE
;Lowest value :

;Highest value :

;Field name :
TEXTID
;Type of field :
NUMBER
;Length of field / Display length :
6/6
;Is this field in the base table Y/N :
Y
;Is this field part of the primary key Y/N :
Y
;Field to copy primary key from :

;Default value :
99
;Page :
7
;Line :
3
;Column :
10
```

```
;Prompt :
TEXT ID
;Display prompt above field Y/N :
N
;Display prompt once for block Y/N :
N
;Allow field to be entered Y/N :
N
;SQL>
SELECT MAX (TEXTID + 1)
INTO TEXTID
FROM TEXTBLOCK

;Message if value not found :
ID  NOT GENERATED BY SYSTEM
;Must value exist Y/N :
Y
;Field name :

;Block name / Description :

%END
```

LIST OF REFERENCES

1. Ross, A.A., Computer Aided Design of Microprocessor-based
    Controllers, PhD. Dissertation, University of California,
    Davis, June 1978, pp. 10-15.

2. McWilliams, Thomas, "SCALD: Structured Computer Aided
    Logic Design", 15th Design Automation Conference
    Proceedings, 1978, Las Vegas, Nevada.

3. Sherlock, Barbara J., User-Friendly, Syntax Directed Input
    to a Computer Aided Design System, Master's Thesis,
    Naval Postgraduate School, Monterey, California, June
    1983, pp. 22-32.

4. Ross, A.A. and Loomis, H.H. Jr., "Computer Aided Design of
    Microprocessor Based Systems", 15th Design Automation
    Conference Proceedings, pp. 227-230, IEEE.

5. Oracle Users Manual, Rlational Systems Incorporated,
    March 1983.

6. Date, C.J., An Introduction to Database Systems,
    Addison-Wesley Publishing Company, pp. 382-384, 1977.

7. Katz, Randy H., "David: Design Aids for VLSI Using
    Integrated Databases", IEEE TC Database Engineering
    Bulletin, Vol. 5, No. 2, June 1982, pp. 29-32.

BIBLIOGRAPHY

Matelan, M.N., "Automating the Design of Dedicated Real Time Control Systems", Preprint UCRL-78651, Lawrence Livermore Lab, 21 August 1976.

INITIAL DISTRIBUTION LIST

No. Copies

1. Defense Technical Information Center   2
   Cameron Station
   Alexandria, Virginia 22314

2. Library, Code 0142   2
   Naval Postgraduate School
   Monterey, California 93943

3. Chairman, Code 52   2
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California 93943

4. LTCOL Alan A. Ross. Code 52Rs   4
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California 93943

5. LT Heather J. Walden, USN   1
   304 Mansion Drive
   Alexandria, Virginia 22302